# Tucuxi CLI Software Usability Specification

***Release 1.0***

**Tucuxi dev team**

**Feb 18, 2022**

# Contents

# 1

# Tucucli specification

The command line tool *tucucli* allows to use all the Tucuxi core functionalities.

The main purpose of *tucucli* is to perform computations. This will return a complete computation for a specific query. Such query embeds information about the patient (covariates), the drug treatment (dosage history + samples concentrations), as well as the specification of the calculation to be done.

The following computations can be performed:

- Single points prediction
- Concentration prediction at time of samples
- Full concentration prediction
    - Typical patient
    - A priori prediction (with the patient's covariates)
    - A posteriori prediction (with covariates and samples)
- Percentiles computation
    - With typical patient PK parameters
    - A priori percentiles (with the patient's covariates)
    - A posteriori percentiles (with covariates and samples)
- Suggestions of dosage adjustments

All required information of a computation is given thanks to an XML format. The details of this format are here: *Query Format*.

The global structure of the XML file is a *tucuxiComputation*. More details on page *Query Response Format*.

## 1.1 Running Tucucli

*tucucli* has a very simple interface: It can have five arguments:

1. The path where to find the drug model files
2. The path to the file containing the query
3. The path to the output file (not yet created)
4. An optional path to a directory in which query files will be stored
5. An optional path to a directory in which csv files (containing only numericals values) will be stored

```
Usage:
  ./tucucli [OPTION...]

  -d, --drugpath arg      Drug files path
  -i, --input arg         Input request file
  -o, --output arg        Output response file
  -q, --querylogpath arg  Query folder path
      --csv arg           Csv file path (.dat)
      --help              Print help
```

The drug files path is not mandatory, and has a default value which is <path_to_tucucli>/drugs2 .

The query folder path is not mandatory.

Both the input request file and the output response folder are mandatory

Some logs are sent to the standard output, and allow to get some

Data file path is not mandatory. If argument not filled, data file won't be created.

### 1.1.1 Return value

The return value of the tucucli process can be any of the following:

Table 1: Return value

| Value | Description |
|---|---|
| 0 | Everything went well |
| 1 | Some requests are wrong, but some are good |
| 2 | None of the requests could be calculated correctly |
| 3 | Error during xml query import |
| 4 | Xml file or string with bad format (not imported yet) |
| 5 | Undefined error (should not happen) |
| -1 | Error in the format of arguments |
| -2 | Missing argument -i and/or -o |

If the computation is partially or fully correct, then the response file will contain information about what happened.

# 2 Query format

**Contents**



This chapter presents the way computing queries have to be described. The input format is valid for both the command line interface and the REST server.

The global structure of the XML file is the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<query>
  <queryId>123_ch.heig-vd.Tucuxi.gentamincin_neonates</queryId>
  <clientId>123</clientId>
  <date>28.04.2020</date>
  <language>en</language>
  <drugTreatment>
    <patient>
      <covariates></covariates>
    </patient>
    <drugs></drugs>
  </drugTreatment>
  <requests></requests>
</query>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <query> | | 1:1 | Data about the query |
| __<queryId> | string | 1:1 | Query ID |
| __<clientId> | string | 1:1 | Client ID |
| __<date> | date | 1:1 | Query date |
| __<language> | string | 1:1 | Languages specifications |
| __<drugTreatment> | | 1:1 | Drug treatment specifications |
| ____<patient> | | 1:1 | Patient specifications |
| ______<covariates> | *Covariates* | 1:1 | Covariates specifications |
| ____<drugs> | *Drugs* | 1:1 | Drugs specifications |
| __<requests> | *Requests* | 1:1 | Requests specification |

The XML format is described in the file `computing_query.xsd` which is used by Tucuxi in order to check the structure correctness before loading the query description file.

The <language> tag is an string enumerate. It can be “en”, “fr”, “de” or “it”, but currently this tag is not used by the computing engine.

## 2.1 Covariates

The covariates of the file is described inside the <covariates> tag. A standard covariates looks like this:

```
<covariates>
  <covariate>
    <covariateId>height</covariateId>
    <date>26.01.1998</date>
    <value>180</value>
    <unit>cm</unit>
    <dataType>double</dataType>
    <nature>continuous</nature>
  </covariate>
</covariates>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <covariate> | | 0:$\infty$ | Description of a covariate |
| __<covariateId> | string | 1:1 | The covariate’s unique identifier |
| __<date> | date | 1:1 | The covariate’s measure date |
| __<value> | string | 1:1 | The covariate’s value |
| __<unit> | string | 1:1 | The covariate’s unit |
| __<dataType> | string | 1:1 | The covariate’s value data type |
| __<nature> | string | 0:1 | The covariate’s nature |

A covariate ID must identify the covariate. The <date> refer to the date of the covariate measurement.

The <unit> of the covariate is the unit presented to the user - for example kg for the covariate weight. The <dataType>, on the other hand, is the internal type used to store the covariate's value. It can be either an int, a double or a boolean. In case of boolean, then the term is bool, and the value shall be 1 for true and 0 for false.

The <dataType> tag is an string enumerate. It can be "int", "double", "bool" or "date". Typically, a birthdate would be of type "date".

**Warning:** If the drug model deals with the age, then a birthdate instead of an age has to be given, as it is used to calculate the age.

The <nature> tag is a string enumerate. It cab be "continuous", "discrete" or "categorical". Standard covariates are "continuous", and currently this field is not used by the computing engine.

## 2.2 Drugs

The drugs embed information about various drug treatments the patient could be in. The *drugId* is then referenced by the computing requests, and the computing engine gets the information from the corresponding drug treatment. A standard *drugs* looks like this:

```
<drugs>
  <drug>
    <drugId>gentamincin</drugId>
    <activePrinciple>vancomycin</activePrinciple>
    <brandName>somebrand</brandName>
    <atc>J01XA01</atc>
    <treatment></treatment>
    <samples></samples>
    <targets></targets>
  </drug>
</drugs>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <drug> | | 0:∞ | Description of a drug |
| __<drugId> | string | 1:1 | The drug's unique id |
| __<activePrinciple> | string | 1:1 | The drug's active principle |
| __<brandName> | string | 1:1 | The drug's brand name |
| __<atc> | string | 1:1 | The drug's atc |
| __<treatment> | *Treatment* | 1:1 | The drug's treatment |
| __<samples> | *Samples* | 0:1 | The drug's samples |
| __<targets> | *Targets* | 0:1 | The drug's targets |

The *drugId* uniquely identifies the drug. It should be equal to the *drugId* of the selected drug models.

The *activePrinciple* uniquely identifies the active principle of the drug.

The *brandName* is currently unused.

The *ATC* is currently unused but could serve to double check that the drug model implements the correct drug.

The tags <treatment>, <samples> and <targets> are developped in chapter below.

## 2.3 Treatment

The Treatment of the file is described inside the <treatment> tag. It contains the dosage history of the patient for a specific drug.

A standard treatment looks like this:

```
<treatment>
  <dosageHistory>
    <dosageTimeRange>
      <start>2018-01-09T14:29:00</start>
      <end>2018-01-10T01:53:00</end>
      <dosage></dosage>
    </dosageTimeRange>
  </dosageHistory>
</treatment>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageHistory> | | 0:1 | History of dosages |
| __<dosageTimeRange> | | 0:∞ | Dosage specifications |
| ____<start> | date | 1:1 | The dosage's start date |
| ____<end> | date | 1:1 | The dosage's end date |
| ____<dosage> | *Query dosage* | 0:∞ | Dosage specifications |

The dosage specifications are described *here*.

## 2.4 Samples

The samples are described inside the <samples> tag. It is a list of drug concentration measurements.

A standard samples looks like this:

```
<samples>
  <sample>
    <sampleId>1</sampleId>
    <sampleDate>2018-01-10T14:41:54</sampleDate>
    <concentrations>
      <concentration>
        <analyteId>gentamincin</analyteId>
        <value>23</value>
        <unit>mg/l</unit>
      </concentration>
    </concentrations>
  </sample>
</samples>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <sample> | | 1:∞ | Description of a sample |
| __<sampleId> | int | 1:1 | The sample's unique id |
| __<sampleDate> | date | 1:1 | The sample's date |
| __<concentrations> | | 1:1 | Concentrations |
| ____<concentration> | | 0:∞ | Concentration specification |
| ______<analyteId> | string | 1:1 | Analyte unique id |
| ______<value> | decimal | 1:1 | Concentration's value |
| ______<unit> | string | 1:1 | Concentration's unit |

A sample contain an ID, a date and one or more concentration.

For one sample there can be more than one concentration, as the drug could be composed of multiple analytes. So, for each concentration the *analyteId* identifies the analyte for which the concentration has been measured.

## 2.5 Targets

The targets are used by the adaptation engine to fit the dosage to the specific needs of the patient. A drug model embeds such targets, but it is possible to override them by supplying new targets.

A standard targets looks like this:

```
<targets>
  <target>
    <activeMoietyId>imatinib</activeMoietyId>
    <targetType>residual</targetType>
    <unit>mg</unit>
    <min>20</min>
    <best>25</best>
    <max>30</max>
    <inefficacyAlarm>15</inefficacyAlarm>
    <toxicityAlarm>50</toxicityAlarm>
    <mic></mic>
  </target>
</targets>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <target> | | 1:∞ | List of targets |
| __<activeMoietyId> | string | 1:1 | active moiety for this target |
| __<targetType> | string | 1:1 | Target's type |
| __<unit> | string | 1:1 | Target's unit |
| __<min> | decimal | 1:1 | Target's minimum value |
| __<best> | decimal | 1:1 | Target's best value |
| __<max> | decimal | 1:1 | Target's maximum value |
| __<tMin> | decimal | 0:1 | Target's minimum time, for peaks |
| __<tBest> | decimal | 0:1 | Target's best time, for peaks |
| __<tMax> | decimal | 0:1 | Target's maximum time, for peaks |
| __<inefficacyAlarm> | decimal | 0:1 | Target's inefficacy alarm value |
| __<toxicityAlarm> | decimal | 0:1 | Target's toxicity alarm value |
| __<mic> | mic | 0:1 | Minimum inhibitory concentration |

The tags <tMin>, <tBest>, and <tMax> are only required in case of a target mentionning a peak: **peak** or **peakDividedByMic**.

The mic is defined as:

```
<mic>
  <unit>ug/l</unit>
  <micValue>1.0</micValue>
</mic>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <mic> | | 0:1 | A MIC field |
| __<unit> | string | 1:1 | MIC's unit |
| __<micValue> | decimal | 1:1 | MIC's value |

For more information about targets, please have a look here.

## 2.6 Requests

The *drugTreatment* contains all information known about the patient (dosage history, samples, covariates). All computations are performed based on this knowledge.

The requests represent what kind of computation needs to be done on what drug. A query can be composed of many requests.

The Requests of the file are described inside the <requests> tag. A standard Requests looks like this:

```
<requests>
  <request>
    <requestId></requestId>
```

(continues on next page)

(continued from previous page)

```
    <drugId></drugId>
    <drugModelId></drugModelId>
    <COMPUTING_TRAIT></COMPUTING_TRAIT>
  <request>
</requests>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <request> | | 1:∞ | Request specifications |
| __<requestId> | string | 1:1 | Request's unique id |
| __<drugId> | string | 1:1 | Drug's unique id |
| __<drugModelId> | string | 1:1 | Drug model unique id |
| __<COMPUTING_TRAIT> | | 1:1 | This tag doesn't exist, see warning below |

The *requestId* uniquely identifies the request. It shall be unique within a query.

The *drugId* identifies the drug for which the computation has to be performed. The engine will look for that drug in the first part of the XML to extract the corresponding information (samples, dosages).

The *drugModelId* identifies the drug model to be used for the computation. It has to be a valid one, typically retrieved from the list of available drug models. Obviously it should be compatible with the *drugId*, else no computation can be performed.

The last tag is a choice between the various computation possibilities. Each type of computation then embeds the required information specific to it. Each trait will have a *computingOption* field, and some other specific fields.

**Warning:** The tag <COMPUTING_TRAIT> only exist for the representation above. This tag must be replaced by one of the following tags : predictionTraits, predictionAtTimesTraits, predictionAtSampleTimesTraits, percentilesTraits or adjustmentTraits

### 2.6.1 PredictionTraits

A prediction trait is used compute a prediction over a certain period of time.

For instance:

```
<predictionTraits>
  <computingOption></computingOption>
  <nbPointsPerHour>20</nbPointsPerHour>
  <dateInterval>
    <start>2018-01-12T07:00:00</start>
    <end>2018-03-15T12:59:00</end>
  </dateInterval>
</predictionTraits>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <predictionTraits> | | 1:∞ | Prediction Traits specifications |
| __<computingOption> | *ComputingOption* | 1:1 | computing options |
| __<nbPointsPerHour> | int | 0:1 | Number of points per hour |
| __<dateInterval> | | 0:1 | Date interval specifications |
| ____<start> | date | 1:1 | Start date |
| ____<end> | date | 1:1 | End date |

The number of points per hour defines the density of the resulting concentrations. Internally the software checks whether there are not too many points to calculate. If this is the case, then no computation is performed and a status code *TooBig* is returned.

The <computingOption> tag is developped *here*.

### 2.6.2 PredictionAtTimesTraits

This type of traits allows to specify specific times at which we need a concentration calculation.

```
<predictionAtTimesTraits>
  <computingOption></computingOption>
  <dates>
    <date></date>
  </dates>
</predictionAtTimesTraits>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <predictionAtTimes-Traits> | | 1:∞ | Prediciton at times traits specifications |
| __<computingOption> | *ComputingOption* | 1:1 | computing options |
| __<dates> | | 1:1 | Dates specifications |
| ____<date> | date | 0:∞ | Actual date |

The <computingOption> tag is developped *here*.

### 2.6.3 PredictionAtSampleTimesTraits

This trait asks for concentration predictions at the time of each sample.

```
<predictionAtSampleTimesTraits>
  <computingOption></computingOption>
</predictionAtSampleTimesTraits>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <predictionAtSample-TimesTraits> | | 1:∞ | Prediction at sample times traits specifications |
| __<computingOption> | *ComputingOption* | 1:1 | computing options |

The <computingOption> tag is developped *here*.

### 2.6.4 PercentilesTraits

This trait is meant to be used to calculate percentiles. As for a single prediction, it requires a number of points per hour, a start and an end date. Moreover, a list of percentiles ranks define what percentiles have to be calculated.

For instance the following example asks 5 percentiles (10, 25, 50, 75, 90):

```
<percentilesTraits>
  <computingOption></computingOption>
  <nbPointsPerHour>20</nbPointsPerHour>
  <dateInterval>
    <start>2018-01-12T07:00:00</start>
    <end>2018-03-15T12:59:00</end>
  </dateInterval>
  <ranks>
    <rank>10</rank>
    <rank>25</rank>
    <rank>50</rank>
    <rank>75</rank>
    <rank>90</rank>
  </ranks>
</percentilesTraits>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <percentilesTraits> | | 1:∞ | Percentile traits specifications |
| __<computingOption> | *ComputingOption* | 1:1 | Computing options |
| __<nbPointsPerHour> | int | 0:1 | Number of points on graph |
| __<dateInterval> | | 0:1 | Date interval specifications |
| ____<start> | date | 1:1 | Start date |
| ____<end> | date | 1:1 | End date |
| __<ranks> | | 1:1 | Percentile ranks |
| ____<rank> | double | 1:∞ | Percentile rank number |

The <computingOption> tag is developped *here*.

### 2.6.5 AdjustmentTraits

This trait is used to ask for a dosage adjustment.

The response greatly depends on the adjustment options, but will mainly consist in a list of potential adjustements, being a dosage history and calculated points. These calculated points are the one in the period defined in the *dateInterval*.

Here is an example:

```
<adjustmentTraits>
  <computingOption></computingOption>
  <nbPointsPerHour>20</nbPointsPerHour>
  <dateInterval>
    <start>2018-01-12T07:00:00</start>
    <end>2018-03-15T12:59:00</end>
  </dateInterval>
  <adjustmentDate></adjustmentDate>
  <options>
    <bestCandidatesOption>bestDosage</bestCandidatesOption>
    <loadingOption>noLoadingDose</loadingOption>
    <restPeriodOption>noRestPeriod</restPeriodOption>
    <steadyStateTargetOption>atSteadyState</steadyStateTargetOption>
    <targetExtractionOption>populationValues</targetExtractionOption>
    <formulationAndRouteSelectionOption>lastFormulationAndRoute</
↪formulationAndRouteSelectionOption>
  </options>
</adjustmentTraits>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <adjustmentTraits> | | 1:∞ | Adjustement traits specifications |
| __<computingOption> | *ComputingOption* | 1:1 | General computing options |
| __<nbPointsPerHour> | int | 0:1 | Number of points on graph |
| __<dateInterval> | | 0:1 | Date interval specifications |
| ____<start> | date | 1:1 | Start date |
| ____<end> | date | 1:1 | End date |
| __<adjustmentDate> | date | 0:1 | Date of adjustment |
| __<options> | | 1:1 | Options specifications |
| ____<bestCandidatesOption> | string | 1:1 | What candidates are retrieved |
| ____<loadingOption> | string | 1:1 | loading dose or not |
| ____<restPeriodOption> | string | 1:1 | restint period or not |
| ____<steadyStateTargetOption> | string | 1:1 | computation at steady state or not |
| ____<targetExtractionOption> | string | 1:1 | Extraction option for targets |
| ____<formulationAndRouteSelectionOption> | string | 1:1 | Selection of the potential formulations and routes |

The <computingOption> tag is developped *here*.

The <bestCandidatesOption> tag is an string enumerate. It can be “bestDosage”, “allDosages” or “bestDosagePerInterval”.

| bestCandidatesOption | Description |
|---|---|
| bestDosage | Only the best dosage is sent back in the response |
| allDosages | All dosages that are within the [min,max] range are sent back |
| bestDosagePerInterval | For each possible interval only the best dosage is sent back |

The <loadingOption> tag is an string enumerate. It can be “noLoadingDose” or “loadingDoseAllowed”.

| loadingOption | Description |
|---|---|
| noLoadingDose | No loading dose can be added to the new dosage |
| loadingDoseAllowed | If the current dosage is under the target, a loading dose can be added at the beginning of the new dosage to more rapidly reach the optimum |

The <restPeriodOption> tag is an string enumerate. It can be “noRestPeriod” or “restPeriodAllowed”.

| restPeriodOption | Description |
|---|---|
| noRestPeriod | No rest period can be added to the new dosage |
| restPeriodAllowed | If the current dosage is over the target, a rest period can be added at the beginning of the new dosage to more rapidly reach the optimum |

The <steadyStateTargetOption> tag is an string enumerate. It can be “atSteadyState” or “withinTreatmentTimeRange”.

| steadyStateTargetOption | Description |
|---|---|
| atSteadyState | The targets are evaluated at steady state |
| withinTreatmentTimeRange | The targets are evaluated within the treatment range. Typically useful when the treatment duration is fixed, and for cumulative AUC targets. |

The <targetExtractionOption> tag is an string enumerate. It can be “populationValues”, “aprioriValues”, “individualTargets”, “individualTargetsIfDefinitionExists”, “definitionIfNoIndividualTarget” or “individualTargetsIfDefinitionExistsAndDefinitionIfNoIndividualTarget”.

| targetExtractionOption | Description |
|---|---|
| populationValues | Forces the population values to be used |
| aprioriValues | Forces the a priori values to be calculated and used |
| individualTargets | Only use the individual targets |
| individualTargetsIfDefinitionExists | Only use the individual targets if a target definition exists |
| definitionIfNoIndividualTarget | Use the individual target, and if for an active moiety and a target type no individual target exists, then use the definition |
| individualTargetsIfDefinitionExistsAndDefinitionIfNoIndividualTarget | Use the individual target if a target definition exist, and if for an active moiety and a target type no individual target exists, then use the definition |

The <formulationAndRouteSelectionOption> tag is an string enumerate. It can be “lastFormulationAndRoute”, “defaultFormulationAndRoute” or “allFormulationAndRoutes”.

| formulationAndRouteSelectionOption | Description |
|---|---|
| lastFormulationAndRoute | Use only the last formulation and route used in the current treatment. If the treatment is empty, then use the default formulation and route of the drug model. |
| defaultFormulationAndRoute | Use only the default formulation and route of the drug model |
| allFormulationAndRoutes | Use all available formulation and routes of the drug model |

### 2.6.6 ComputingOption

The *ComputingOption* field is common to all computing traits. It defines what type of PK parameters to use, and what information should be sent back with the response.

Here is an example:

```
<computingOption>
  <parametersType>aposteriori</parametersType>
  <compartmentOption>allActiveMoieties</compartmentOption>
  <retrieveStatistics>true</retrieveStatistics>
  <retrieveParameters>true</retrieveParameters>
  <retrieveCovariates>true</retrieveCovariates>
</computingOption>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <computingOption> | | 1:1 | Computing option specifications |
| __<parametersType> | string | 1:1 | Parameters type |
| __<compartmentOption> | string | 0:1 | Compartment option |
| __<retrieveStatistics> | bool | 1:1 | If yes, then statistics for all dosage intervals are retrieved |
| __<retrieveParameters> | bool | 1:1 | If yes, then PK parameters are retrieved for all dosage intervals |
| __<retrieveCovariates> | bool | 1:1 | If yes, then covariates values are retrieved for all dosage intervals |

The <parametersType> option defines what parameters should be used for computation. It can be “population”, “apriori”, “aposteriori”.

| parametersType | Description |
|---|---|
| population | Use the typical patient parameters, without any information about the patient |
| apriori | Use a priori parameters, updated thanks to the patient covariates |
| aposteriori | Use a posteriori parameters, updated thanks to the patient covariates and the available drug concentration measurements |

The <compartmentOption> tag is an string enumerate. It can be “allActiveMoieties”, “allAnalytes”, “allCompartments” or “specific”.

| compartmentOption | Description |
|---|---|
| allActiveMoieties | Retrieve concentrations for all active moieties (the default that should be used) |
| allAnalytes | Retrieve concentrations for all analytes (and all active moieties) |
| allCompartments | Retrieve concentrations for all compartments (not yet implemented) |
| specific | Not yet supported |

# 3 Query dosage

**Contents**



A dosage history is a list of dosage time ranges. Each range has a start and end date, and is composed of a dosage. The dosage subsystem is extremely flexible, and allows to describe dosages such that "take a drug at 8:00 every day, except on Sunday".

This flexibility comes with a quite complex management of the XML structure.

**Warning:** For each specific dosage where a formulation and route has to be given, this formulation and route shall be identical to the one proposed by the selected drug model.

The dosage of a dosage time range is described inside the <dosage> tag. A standard dosage looks like this:

```
<dosage>
  <dosageSteadyState>
    <lastDoseDate></lastDoseDate>
    <DOSAGE_CHOICE></DOSAGE_CHOICE>
  </dosageSteadyState>
  or
  <dosageLoop>
    <DOSAGE_CHOICE></DOSAGE_CHOICE>
  </dosageLoop>
  or
  <DOSAGE_CHOICE></DOSAGE_CHOICE>
</dosage>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageSteadyState> | | 1:1 | Steady state (without a starting time) |
| <dosageLoop> | | 1:1 | Dosage treatment without time limit |
| <DOSAGE_CHOICE> | | 1:$\infty$ | This tag doesn't exist, see warning below |

A dosage contains only one of the 2 proposed tags at the same time.

The dosageSteadyState is used to represent the fact that the dosage already reached steady state. It acts like a dosageLoop, except it requires a date of a specific dose.

The dosageLoop is used when the dosage is repeated continually (without limit of time).

The DOSAGE_CHOICE is used when the dosage has a duration.

**Warning:** The tag <DOSAGE_CHOICE> only exist for the representation above. This tag must be replaced by one of the following tags : dosageRepeat, dosageSequence, lastingDosage, dailyDosage or weeklyDosage.

## 3.1 DosageSteadyState

A DosageSteadyState represents a dosage at steady state. It is defined by a last dose (that can be anywhere in time), and a dosageLoop that has been repeated to reach the steady state.

```
<dosageSteadyState>
  <lastDoseDate>2018-07-07T20:00:00</lastDoseDate>
  <DOSAGE_CHOICE></DOSAGE_CHOICE>
</dosageSteadyState>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageSteadyState> | | 1:1 | Dosage Repeat specifications |
| __<lastDoseDate> | xs:datetime | 1:1 | Date and time of a dose |
| __<DOSAGE_CHOICE> | | 1:$\infty$ | This tag doesn't exist, see warning below |

See Dosage warning for <DOSAGE_CHOICE> tag.

## 3.2 DosageRepeat

A DosageRepeat represents a repeated dosage. This is used when the dosage must be taken a defined number of times.

```
<dosageRepeat>
  <iterations>4</iterations>
  <DOSAGE_CHOICE></DOSAGE_CHOICE>
</dosageRepeat>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageRepeat> | | 1:1 | Dosage Repeat specifications |
| __<iterations> | int | 1:1 | Number of repetitions |
| __<DOSAGE_CHOICE> | | 1:∞ | This tag doesn't exist, see warning below |

The iteration tag corresponds to the the number of dosing repetitions.

See Dosage warning for <DOSAGE_CHOICE> tag.

## 3.3 DosageSequence

A DosageSequence is composed of a series of bounded dosages.

```
<dosageSequence>
  <DOSAGE_CHOICE></DOSAGE_CHOICE>
  <DOSAGE_CHOICE></DOSAGE_CHOICE>
</dosageSequence>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageSequence> | | 1:1 | Dosage Sequence specifications |
| __<DOSAGE_CHOICE> | | 1:∞ | This tag doesn't exist, see warning below |

See Dosage warning for <DOSAGE_CHOICE> tag.

## 3.4 LastingDosage

This type of dosage provides a dose and an interval. It is typically used within a dosageLoop or a dosageRepeat.

```
<lastingDosage>
  <interval>24:00:00</interval>
  <dose>
    <value>550</value>
    <unit>mg</unit>
    <infusionTimeInMinutes>600</infusionTimeInMinutes>
  </dose>
  <formulationAndRoute>
    <formulation>parenteralSolution</formulation>
    <administrationName>administrationName</administrationName>
    <administrationRoute>intramuscular</administrationRoute>
    <absorptionModel>intravascular</absorptionModel>
  </formulationAndRoute>
</lastingDosage>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <lastingDosage> | | 1:1 | Lasting dosage specifications |
| __<interval> | time | 1:1 | Dosage interval |
| __<dose> | | 1:1 | Description of a dose |
| ____<value> | decimal | 1:1 | Dose's value |
| ____<unit> | string | 1:1 | Doses's unit |
| ____<infusionTimeIn-Minutes> | decimal | 0:1 | Doses's infusion time (min) |
| __<formulationAn-dRoute> | | 1:1 | Description of a formulation and route |
| ____<formulation> | string | 1:1 | Formulation |
| ____<administra-tionName> | string | 1:1 | Name of administration |
| ____<administra-tionRoute> | string | 1:1 | Way of administration |
| ____<absorptionModel> | string | 1:1 | Way of absorbing |

The <formulation> tag is an string enumerate. It can be "undefined", "parenteralSolution", "oralSolution".

The <administrationRoute> tag is an string enumerate. It can be "undefined", "intramuscular", "intravenousBolus", "intravenousDrip", "nasal", "oral", "rectal", "subcutaneous", "sublingual", "transdermal" or "vaginal".

The <absorptionModel> tag is an string enumerate. It can be "undefined", "intravascular", "extravascular", "extravascularLag" or "infusion".

## 3.5 DailyDosage

This type of dosage provides a dose and a time in the day. This is used on daily dosage. It can be used to say that the patient takes the drug every day at 8:00.

```
<dailyDosage>
  <time>10:00:00</time>
  <dose>
    <value>550</value>
    <unit>mg</unit>
    <infusionTimeInMinutes>600</infusionTimeInMinutes>
  </dose>
  <formulationAndRoute>
    <formulation>parenteralSolution</formulation>
    <administrationName>something</administrationName>
    <administrationRoute>transdermal</administrationRoute>
    <absorptionModel>infusion</absorptionModel>
  </formulationAndRoute>
</dailyDosage>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dailyDosage> | | 1:1 | Daily dosage specifications |
| __<time> | time | 1:1 | Hour of dosage |
| __<dose> | | 1:1 | Description of a dose |
| ____<value> | decimal | 1:1 | Dose's value |
| ____<unit> | string | 1:1 | Doses's unit |
| ____<infusionTimeIn-Minutes> | decimal | 0:1 | Doses's infusion time (min) |
| __<formulationAn-dRoute> | | 1:1 | Description of a formulation and route |
| ____<formulation> | string | 1:1 | Formulation |
| ____<administra-tionName> | string | 1:1 | Administration name |
| ____<administra-tionRoute> | string | 1:1 | Way of administration |
| ____<absorptionModel> | string | 1:1 | Way of absorbing |

The <formulation> tag is an string enumerate. It can be "undefined", "parenteralSolution", "oralSolution".

The <administrationRoute> tag is an string enumerate. It can be "undefined", "intramuscular", "intravenousBolus", "intravenousDrip", "nasal", "oral", "rectal", "subcutaneous", "sublingual", "transdermal" or "vaginal".

The <absorptionModel> tag is an string enumerate. It can be "undefined", "intravascular", "extravascular", "extravascularLag" or "infusion".

## 3.6 WeeklyDosage

This type of dosage provides a dose, a day in the week and a time in the day. It is meant to express the idea of a dose taken once a week.

```
<weeklyDosage>
  <day>2</day>
  <time>11:00:00</time>
  <dose>
    <value>550</value>
    <unit>mg</unit>
    <infusionTimeInMinutes>600</infusionTimeInMinutes>
  </dose>
  <formulationAndRoute>
    <formulation>parenteralSolution</formulation>
    <administrationName>someName</administrationName>
    <administrationRoute>transdermal</administrationRoute>
    <absorptionModel>infusion</absorptionModel>
  </formulationAndRoute>
</weeklyDosage>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <weeklyDosage> | | 1:1 | Weekly dosage specifications |
| __<day> | int | 1:1 | Day of dosage (between 0 and 6) |
| __<time> | time | 1:1 | Hour of dosage |
| __<dose> | | 1:1 | Description of a dose |
| ____<value> | decimal | 1:1 | Dose's value |
| ____<unit> | string | 1:1 | Doses's unit |
| ____<infusionTimeInMinutes> | decimal | 0:1 | Doses's infusion time (min) |
| __<formulationAndRoute> | | 1:1 | Description of a formulation and route |
| ____<formulation> | string | 1:1 | Formulation |
| ____<administrationName> | string | 1:1 | Administration name |
| ____<administrationRoute> | string | 1:1 | Way of administration |
| ____<absorptionModel> | string | 1:1 | Way of absorbing |

The <formulation> tag is an string enumerate. It can be "undefined", "parenteralSolution", "oralSolution".

The <administrationRoute> tag is an string enumerate. It can be "undefined", "intramuscular", "intravenousBolus", "intravenousDrip", "nasal", "oral", "rectal", "subcutaneous", "sublingual", "transdermal" or "vaginal".

The <absorptionModel> tag is an string enumerate. It can be "undefined", "intravascular", "extravascular", "extravascularLag" or "infusion".

# 4 Query response format

**Contents**



This chapter presents the format of the computing query responses. This format is valid for both the command line interface and the REST server.

## 4.1 Structure

The global structure of the XML response is the following:

```
<tucuxiComputation xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
 xsi:noNamespaceSchemaLocation='computing_response.xsd'>
  <queryId>ch.tucuxi.tobramycin_test</queryId>
  <queryStatus></queryStatus>
  <responses></responses>
</tucuxiComputation>
```

Table 1: Query response content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <queryId> | string | 1:1 | Query unique identifier |
| <queryStatus> | *queryStatus* | 1:1 | Query Status container |
| <responses> | *Responses* | 1:1 | Query responses |

## 4.2 Status

The status is a structure that contains all status information.

```
<STATUS_TYPE>
  <statusCode>0</statusCode>
  <statusCodeLit>Ok</statusCodeLit>
  <message>everything ok</message>
  <description>description</description>
</STATUS_TYPE>
```

Table 2: status structure content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <statusCode> | int | 1:1 | Status code identifier |
| <statusCodeLit> | string | 1:1 | Status signification |
| <message> | string | 0:1 | Status message |
| <description> | string | 0:1 | Status description |

**Warning:** The tag <STATUS_TYPE> only exist for the representation above. It can be either <queryStatus> or <requestStatus>, depending on where it is used.

For status code details, please refer to *QueryStatus* and *RequestStatus*.

## 4.3 Responses

The responses of are described inside the <responses> tag. A standard responses looks like this:

```
<responses>
  <response>
    <requestId>request</requestId>
    <requestStatus></requestStatus>
    <requestType>adjustment</requestType>
    <COMPUTING_TRAIT></COMPUTING_TRAIT>
  </response>
</responses>
```

Table 3: Response content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <response> | | 0:$\infty$ | Response specifications |
| __<requestId> | string | 1:1 | Request unique Id |
| __<requestStatus> | *requestStatus* | 1:1 | RequestStatus container |
| __<requestType> | string | 1:1 | The type of request (see below) |
| __<DATA_RESPONSE> | | 1:1 | This tag doesn't exist see warning below |

**Warning:** The tag <DATA_RESPONSE> only exist for the representation above. Its value depends on the type of the request.

<DATA_RESPONSE> can be one of these tags : *dataAdjustement*, *dataPrediction*, *dataPoints* and *dataPercentiles*.

<requestType> can be one of the these values : adjustment, prediction, singlePoints and percentiles.

### 4.3.1 DataPrediction

DataPrediction is used when computing a prediction on a time range.

```
<dataPrediction>
  <analyteIds>
    <analyteId></analyteId>
  </analyteIds>
  <cycleDatas></cycleDatas>
</dataPrediction>
```

Table 4: DataPrediction content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <analyteIds> | | 1:1 | Analyte id container |
| __<analyteId> | string | 0:∞ | Analyte id's |
| __<cycleDatas> | *CycleDatas* | 1:1 | Cycle Datas |

For each analyte, the analyte Id is retrieved, as well as all the predicted concentrations, embedded in a *cycleData*.

### 4.3.2 DataPoints

DataPoints is used when single points predictions or prediction at sample times are performed. The date is quite simple and consists of a list of points. Each point is a tuple time-value.

```
<dataPoints>
  <unit></unit>
  <points>
    <point>
      <time></time>
      <value></value>
    </point>
  </points>
</dataPoints>
```

Table 5: DataPoints content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <unit> | unit | 1:1 | DataPoints unit |
| <points> | | 1:1 | Points specifications |
| __<point> | | 1:∞ | Point specifications |
| ____<time> | date | 1:1 | Point's time |
| ____<value> | concentration | 1:1 | Point's value |

### 4.3.3 DataPercentiles

DataPercentiles is used when percentiles has been calculated.

Here is an example:

```
<dataPercentiles>
  <percentile>
    <rank></rank>
    <cycleData></cycleData>
  </percentile>
</dataPercentiles>
```

Table 6: DataPercentiles content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <percentile> | | 0:∞ | Percentile |
| __<rank> | double | 1:1 | rank |
| __<cycleDatas> | *CycleDatas* | 1:1 | Cycle Datas |

Each percentile is represented by its rank [0,100], and by a list of cycle data that embed the prediction curve for that percentile.

### 4.3.4 DataAdjustement

During an adjustment the computing engine tries to find the best dosage for the specific patient. Many options are used by this engine, but the output is similar for each option. It consists of all adjustment candidates (depending on the options), and for each candidate a dosage is proposed and a prediction is calculated.

Here is an exemple of this structure:

```
<dataAdjustment>
  <analyteIds>
```

(continues on next page)

(continued from previous page)

```
    <analyteId></analyteId>
  </analyteIds>
  <adjustments>
    <adjustment>
      <score>0.972919</score>
      <targetEvaluations></targetEvaluations>
      <dosageHistory></dosageHistory>
      <cycleDatas></cycleDatas>
    </adjustment>
  </adjustments>
</dataAdjustment>
```

Table 7: DataAdjustement content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <analyteIds> | | 1:1 | Analyte id container |
| __<analyteId> | string | 0:∞ | Analyte id's |
| <adjustments> | | 1:1 | Adjustment container |
| __<adjustment> | | 1:∞ | Adjustement description |
| ____<score> | double | 1:1 | The score of this adjustment (1 is the best) |
| ____<targetEvaluations> | *TargetEvaluations* | 0:1 | Target Evaluations |
| ____<dosageHistory> | *DosageHistory* | 1:1 | Dosage History |
| ____<cycleDatas> | *CycleDatas* | 1:1 | Cycle Datas |

The list of analyte Ids for which the cycleDatas will contain values. Useful if there is more than one analyte in the drug model.

The score is the one calculated for the targets. The best score is 1, and it is used to sort the adjustments.

There is one targetEvaluation per target.

### TargetEvaluations

The TargetEvaluations represents the evaluation of each target, for an adjustment.

Here is an example:

```
<targetEvaluations>
  <targetEvaluation>
    <targetType></targetType>
    <unit></unit>
    <value></value>
    <score></score>
  </targetEvaluation>
</targetEvaluations>
```

Table 8: TargetEvaluations content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <targetEvaluation> | | 1:∞ | TargetEvaluation specification |
| __<targetType> | string | 1:1 | Type of target |
| __<unit> | unit | 1:1 | Target's unit |
| __<value> | double | 1:1 | Target's value |
| __<score> | double | 1:1 | Target's score |

The target type corresponds to the ones of the target definitions.

The value is the calculated value. For instance, in the case of a residual target it will be the calculated concentration.

The score is the score of the target. The best value is 1.

### DosageHistory

The DosageHistory contains the dosage history of the daptation.

```
<dosageHistory>
  <dosageTimeRange>
    <start>2018-10-05T10:00:00</start>
    <end>2018-10-15T10:00:00</end>
    <dosage></dosage>
  </dosageTimeRange>
</dosageHistory>
```

Table 9: DosageHistory content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <dosageTimeRange> | | 0:∞ | Dosage Time Range specifications |
| __<start> | date | 1:1 | The dosage's start date |
| __<end> | date | 1:1 | The dosage's end date |
| __<dosage> | *Query dosage* | 0:∞ | Dosage specifications |

### CycleDatas

Predictions and percentiles are retrieved thanks to cycleDatas. A cycleData embeds all information about a dosage interval, from one intake to the next one.

As example:

```
<cycleDatas>
  <cycleData>
    <start>2018-10-05T10:00:00</start>
    <end>2018-10-05T10:00:24</end>
    <unit>ug/l</unit>
    <parameters>
      <parameter>
        <id></id>
        <value></value>
      </parameter>
    </parameters>
    <covariates>
      <covariate>
        <id></id>
        <value></value>
      </covariate>
    </covariates>
    <times></times>
    <values></values>
    <statistics>
      <mean></mean>
      <auc></auc>
      <auc24></auc24>
      <cumulativeAuc></cumulativeAuc>
      <residual></residual>
      <peak></peak>
    </statistics>
  </cycleData>
<cycleDatas>
```

Table 10: CycleDatas content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <cycleData> | | 0:∞ | CycleData specifications |
| __<start> | date | 1:1 | The cycleData start date |
| __<end> | date | 1:1 | The cycleData end date |
| __<unit> | unit | 1:1 | The cycleData concentration values unit |
| __<parameters> | | 0:1 | PK Parameters used for this cycle |
| ____<parameter> | | 1:∞ | A PK parameter |
| ______<id> | string | 1:1 | Parameter ID |
| ______<value> | double | 1:1 | Parameter value |
| __<covariates> | | 0:1 | Covariates used for this cycle |
| ____<covariate> | | 1:∞ | A covariate |
| ______<id> | string | 1:1 | Covariate ID |
| ______<value> | double | 1:1 | Covariate value |
| __<times> | string | 1:1 | A list of times |
| __<values> | string | 1:1 | A list of concentration values |
| __<statistics> | | 0:1 | Cycle data statistics |
| ____<mean> | double | 0:1 | Mean |
| ____<auc> | double | 0:1 | Area under curve |
| ____<auc24> | double | 0:1 | Area under curve on 24h |
| ____<cumulativeAuc> | double | 0:1 | Cumulative area under curve since the beginning of the treatment |
| ____<residual> | double | 0:1 | Residual concentration |
| ____<peak> | double | 0:1 | Peak concentration |

The field *times* contains a comma-separated list of times, in hours, starting from 0. To get the real time of a concentration, this time should be added to *start*.

The field *values* contains a comma-separated list of concentration values. The number of values matches the number of times.

The PK parameters are the PK parameters used at this cycle, and depends on the type of parameters (population, a priori, a posteriori).

As some covariates can change in time, the list of current covariates can be retrieved as well.

The statistics are calculated on this interval and give some insight that can be useful.

The statistics, PK parameters and covariates are not mandatory, and only present if the computing options asked for their retrieval.

**QueryStatus**

Table 11: Query status code

| Code | Name | Meaning |
|---|---|---|
| 0 | OK | Everything fine |
| 1 | PartiallyOk | Some request are wrong |
| 2 | Error | None of the requests could be calculated correctly |
| 3 | ImportError | Error during xml query import |
| 4 | BadFormat | Xml query has bad format |
| 5 | Undefined | Should not be observed |

**RequestStatus**

Table 12: Request status code

| Code | Name | Meaning |
|---|---|---|
| -1 | Undefined | Should not be observed |
| 0 | Ok | Everything fine |
| 1 | TooBig | Too Big computation required |
| 2 | Aborted | Computation aborted by the external agent |
| 3 | ParameterExtractionError | Error in the extraction of parameters |
| 4 | SampleExtractionError | Error in the extraction of samples |
| 5 | TargetExtractionError | Error in target extraction |
| 6 | InvalidCandidate | The candidate is outside the target range |
| 7 | TargetEvaluationError | There was an internal error during target evaluation |
| 8 | CovariateExtractionError | There was an error during covariate extraction |
| 9 | IntakeExtractionError | There was an error during intake extraction |
| 10 | ErrorModelExtractionError | There was an error during residual error model extraction |
| 11 | UnsupportedRoute | The IntakeToCalculatorAssociator could not find a suitable route |
| 12 | AnalyteConversionError | Error during conversion of analytes |
| 13 | AposterioriPercentilesNoSamplesError | A posteriori percentiles calculation, but no samples |
| 14 | ConcentrationCalculatorNoParameters | The ConcentrationCalculator did not find parameters |
| 15 | BadParameters | Bad paramters in the intake calculator |
| 16 | BadConcentration | Bad concentration detected in the intake calculator |
| 17 | DensityError | Density error in the intake calculator |
| 18 | AposterioriEtasCalculationEmptyOmega | In a posteriori Etas calculation, Omega is empty |
| 19 | AposterioriEtasCalculationNoSquareOmega | In a posteriori Etas calculation, Omega is not a square matrix |
| 20 | ComputingTraitStandardShouldNotBeCalled | ComputingTraitStandard::compute() should not be called |
| 21 | CouldNotFindSuitableFormulationAndRoute | Could not find a suitable formulation and route |

continues on next p

Table 12 – continued from previous page

| Code | Name | Meaning |
|---|---|---|
| 22 | MultipleFormulationAndRoutesNotSupported | Multiple formulation and routes are not yet supported |
| 23 | NoPkModelError | Could not find a suitable Pk Model |
| 24 | ComputingComponentExceptionError | An exception was raised in ComputingComponent |
| 25 | NoPkModels | No Pk Models in the collection |
| 26 | NoComputingTraits | The ComputingTraits sent for computation are nullptr |
| 27 | RecordedIntakesSizeError | The recorded intakes size does not fit the predictions |
| 28 | NoPercentilesCalculation | No percentile calculation was performed |
| 29 | SelectedIntakesSizeError | The selected intakes size is not correct |
| 30 | NoAvailableDose | No available dose |
| 31 | NoAvailableInterval | No available interval |
| 32 | NoAvailableInfusionTime | No available infusion time |
| 33 | NoFormulationAndRouteForAdjustment | No formulation and route available for adjustments |
| 34 | ConcentrationSizeError | The concentration vector has not a correct size |
| 35 | ActiveMoietyCalculationError | Error during calculation of active moiety concentration value |
| 36 | NoAnalytesGroup | There are no analytes groups |
| 37 | IncompatibleTreatmentModel | The drug model is incompatible with the drug treatment |
| 38 | ComputingComponentNotInitialized | The computing component has not been initialized |
| 39 | UncompatibleDrugDomain | The drug domain is uncompatible with the patient covariates |
| 40 | NoSteadyState | The steady state can not be attained with this drug model |
| 41 | AposterioriPercentilesOutOfScopeSamplesError | In a posteriori Etas calculation there is no sample close enough to the inta |

# 5 General elements

## 5.1 Units

The units in Tucuxi consist of a base and a multiplier. For example in *ug/l*, *g/l* is the base, and *u* the multiplier, which express a unit of *micrograms per liter*. The convertion factors are used to convert the data produced by the models of Tucuxi. The molar mass is used to give the user the choice to use moles instead of the default units.

## 5.2 stdAprioriValue

In various parts of the drug model, elements can have a default value, and an apriori value calculated with the help of the patient covariates. Every time such a pattern is used, the element is of type *stdAprioriValue*.

Table 1: stdAprioriValue content

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <standardValue> | double | 1:1 | The default value |
| <aprioriComputation> | *Operation* | 0:1 | The operations to calculate the a priori value |

When such an element is used, the software will use default values for calculation involving the typical patient. In case of a priori and a posteriori calculations, it will try to apply the operations on each of such value, in order to better fit the patient reality.

## 5.3 Operation

The operations are used to modify the values of the parameters in accordance with the patient's covariates. They are used to compute the *a priori* parameters and can be of three types: - SoftFormula - HardFormula - MultiFormula

These three options are mutually exclusive, so the possible styles of operations are:

```
<operation>
  <softFormula>
    <inputs>
      <input>
```

(continues on next page)

(continued from previous page)

```
        <id>bodyweight</id>
        <type>double</type>
      </input>
    </inputs>
    <code><![CDATA[
        return bodyweight < 100.0 and bodyweight > 0.0;
            ]]>
    </code>
  </softFormula>
  <comments/>
</operation>
```

```
<operation>
  <hardFormula>formulaId</hardFormula>
  <comments/>
</operation>
```

```
<operation>
  <multiFormula>
    <softFormula>
      <inputs>
        <input>
          <id>bodyweight</id>
          <type>double</type>
        </input>
      </inputs>
      <code><![CDATA[
          return bodyweight < 100.0 and bodyweight > 0.0;
              ]]>
      </code>
    </softFormula>
    <hardFormula>formulaId</hardFormula>
  </multiFormula>
  <comments/>
</operation>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <operation> | | | Description of an operation |
| ____<Softformula> | | 1:∞ | An Javascript operation |
| _____<inputs> | | 1:1 | The list of required inputs |
| _____<input> | | 1:∞ | An input for the formula |
| _____<id> | | 1:1 | The Id of the required input for the formula |
| _____<type> | | 1:1 | The type of data : double, int or bool |
| _____<code> | *Code* | 1:∞ | The operation formula |
| ____<hardFormula> | string | 0:1 | A hardcoded operation |
| ____<multiFormula> | | 0:1 | A multi-operation formula |
| _____<...> | | 1:∞ | Any of softFormula and hardFormula |
| ____<comments> | | 1:1 | Comments about the operation |

An operation can be used in many elements. For instance it is used in parameters, targets, covariates, in order to calculate a priori values.

A formula can use the value of any global or drug-specific covariate. To do so, you must use the covariate's ID , as shown above with *bodyweight*. You can also use any of the drug's parameters, using its ID followed by the *_population* keyword, as in *V_population*. The formula should simply return a value, nothing else is mandatory.

The formula must always be surrounded by the *<![CDATA[* and *]]* markers. The language used to express the formula is based on Javascript and supports a subset of it. A formula must always return a value.

When an operation can contain a list of formula the computing engine shall try to apply the first formula. If there are missing covariates for such formula, then the second formula is tried, and so on, until a valid formula is found.

The list of inputs is important and should contain all the inputs required by the formula (for a soft formula). The type should obviously be correct. In case the input is a covariate, the type should be the same type as the covariate, and in case of a parameter the type is *double*. The editor does not check if the input list is correct or not, so be careful.

## 5.4 Code

A formula is an operation returning a value, based on some inputs. The content of the element is then a source code in the correct format.

The source code must always be surrounded by the *<![CDATA[* and *]]* markers. The language used to express the formula is based on Javascript and supports a subset of it. A formula must always return a value.

The following mathematical functions are available within scripts:

```
Math.E()
Math.log(a)
Math.log10(a)
Math.exp(a)
Math.pow(a,b)
```

(continues on next page)

(continued from previous page)

```
Math.sqr(a)
Math.sqrt(a)

Math.abs(a)
Math.round(a)
Math.min(a,b)
Math.max(a,b)
Math.range(x,a,b)
Math.sign(a)

Math.PI()
Math.toDegrees(a)
Math.toRadians(a)
Math.sin(a)
Math.asin(a)
Math.cos(a)
Math.acos(a)
Math.tan(a)
Math.atan(a)
Math.sinh(a)
Math.asinh(a)
Math.cosh(a)
Math.acosh(a)
Math.tanh(a)
Math.atanh(a)
```

## 5.5 Comments

Before explaining all the specific fields, a word on comments is required, as the <comments> tag can be found at different places of the file. A comment has the following structure:

```
<comments>
  <comment lang='en'>This work is based on the paper nice_paper, that can be found here.
↪</comment>
  <comment lang='fr'>La description de ce médicament est basée sur ce nice_paper, qui␣
↪peut être trouvé ici.</comment>
</comments>
```

| Tag name | Format | Occ. | Description |
|---|---|---|---|
| <comments> | | 1:1 | List of translated comments |
| __<comment lang='xx'> | string | 0:∞ | Comment for a specific language |

It contains as many <comment> tags as required. Each <comment> tag has an attribute *lang* defining the language of the comment, enabling multi-language support for the description of the medical drugs.

Such function must always be written with the *Math.* as prefix. Example:

```
newValue = Math.pow(aValue, anotherValue) + Math.exp(yetAnotherValue);
```

Structures such as *if/then/else* are supported, as in the following example:

```
if (sex > 0.5) {
  aValue = 12;
}
else {
  aValue = 10;
}
```

# 6 Indices and tables

- genindex
- modindex
- search